Comprehensive Study of various approaches for Document Retrieval

Bachelor Project
in
Computer Science and Engineering
By
Varun Singh Bhadauria (2002143)

Under the guidance of

Prof. Saroj Kaushik
Department of Computer Science and Engineering
Indian Institute of Technology Delhi

Department of Computer Science and Engineering,
Indian Institute of Technology Delhi,
New Delhi-110016, India

Certificate

This is to certify that the thesis titled **Comprehensive Study of various approaches for Document Retrieval** being submitted by **Varun Singh Bhadauria** for the award of **Bachelor of Technology** is a record of bonafide work carried out by him under our guidance and supervision at the **Department of Computer Science & Engineering, Indian Institute of Technology, Delhi**. The work presented in this thesis has not been submitted elsewhere, either in part or full, for the award of any other degree or diploma.

Dr. Saroj Kaushik
Department of Computer Science & Engg.
Indian Institute of Technology, Delhi

Acknowledgments

I am greatly indebted to my supervisor Prof. Saroj Kaushik of IIT Delhi for her invaluable technical guidance and moral support during the course of project. I would also like to thank the staff of AI Laboratory, IIT Delhi for their help.

Abstract

Ever since the advent of computer systems and in particular, the Internet, the amount of information at our disposal has been rising exponentially.This phenomenal growth of data is not only a blessing;the more information is available, the more difficult it becomes to find one's way to the particular piece of information of interest.As a consequence, investigations into old and new techniques for dealing with the extra ordinary flood of data remain topical for information science.

The focus of this thesis is on the comprehensive study of various document retrieval approaches comprising the traditional *Vector Space Model*, the well-known Singular Value Decomposition based *Latent Semantic Indexing*, the lesser known Contextual Network Graph based *Spreading Activation* and the new word space based *Random Indexing*.This thesis also conceptualise and implements a *Contextual Vector* based approach for the same.

These approaches are tested against the **MEDLINE** and **NPL** document collections.This thesis ends with an interpretation of the results, a discussion of both the virtues and vices shown by these techniques on the above data sets.

Contents

List of Figures

1 Overview

When searching for information in a retrieval system, people use different terms to describe their information needs.The retrieval system searches through its databases and returns documents indexed with matching terms.Since a concept can be represented by a variety of terms, users may fail to obtain the information they require.This is called the vocabulary terms. The vocabulary problem occures occurs frequently with the classic *Vector Space*[1]approach to address which many approaches has been proposed in the past.In this thesis we study and test the *Latent Semantic Indexing*[2], *Spreading Activation*[9] and *Random Indexing*[3] approaches on the Medline *corpora* of documents, against the vector space approach.

1.1 Motivation

Although these approaches have been investigated alone previously elsewhere there is no comparative study of these approaches collectively on a same document collection.This thesis aims to fill this gap.This thesis also investigates the use of *Context Distance*[5] as new approach for efficient document retrieval.

1.2 Objectives

This thesis develops an end-to-end document retrieval system based on these approaches.The system is evaluated against the commonly used MEDLINE[6] and NPL[**?**] test collection and the results obtained are discussed.

1.3 Report Organization

- **Chapter 2** presents a description of the steps involved in the construction of a document retrieval system.
- **Chapter 3** presents a description of the classic *Vector Space Model*.
- **Chapter 4** presents a description of *Singular Value Decomposition* based *Latent Semantic Indexing*.
- **Chapter 5** presents a description of *Word Space based Random Indexing* approach.
- **Chapter 6** presents a description of *Contextual Network Graph* based approach of *Spreading Activation*.
- **Chapter 7** presents a description of *Contextual Vector* based approach.
- **Chapter 8** brings up the evaluation strategy.
- **Chapter 9** brings up the experimental results and the discussion.

2 Construction Steps of a Document Retrieval System

2.1 What is a document retrieval system?

Definition 1 ***Document Retrieval*** *is defined as the matching of some stated user query against a set of free-text records.*

These records could be of any type of mainly unstructured text, such as newspaper articles, real estate records or paragraphs in a manual.User queries can range from multi-sentence full descriptions of an information need to a few words. A doument retrieval system find information to given criteria by matching text records (documents) against user queriesand consists of a database of documents, a classification algorithm to build a full text index, and a user interface to access the database.A document retrieval system has two main tasks:

- Find relevant documents to user queries.
- Evaluate the matching results and sort them according to relevance.

The document classification scheme (or *indexing* algorithm) in use determines the nature of the document retrieval system.All the approaches discussed in this thesis differ in this scheme.

2.2 What are the construction steps involved?

In order to identify which terms should be used to index a document collection, these terms need to be identified and stored.We now discuss some of the main steps in extracting these terms:

- **Lexical Analysis**: Lexical Analysis is the process of identifying words from a document.This is achieved by checking each input character in a stream of input characters and identifying characters which indicate the start and finish of words.The definition of word depends on the system. The utility *Lex*[7] is a common method of generating a lexical analyser automatically based on a set of regular expressions and actions.

- **Stop lists**: During the auomatic indexing of documents, candidates index terms are usually compared against a 'stop list', which is a list of very common words (such as 'the' or 'a').These terms are removed as they appear too frequently to be useful as index terms.The standard Stop List used in may studies is the list used by *SMART*[8] system, which contains 571 terms.

- **Stemming**: Stemming involves collapsing morphological variants of the same lexical item into a single root.For example, 'process', 'processing' and 'processed' will have the same root 'process'.When used stemming needs to be applied to both the keyword index and the query terms.The advantage of stemming is that a query on keyword 'processing' will also retrieve documents that contain 'process' and 'processed'.The disadvantage of stemming is that it can return terms which stem to the same root but are not related to each other.

- **Document & Query Representation**: In order to reduce the complexity of the documents and make them easier to handle, the document have to be transformed from the full text version to a document vector which describes the contents of the

document. A definition of a document is that it is made of a joint membership of terms which have various patterns of occurrence. These occurrence patterns are often disregarded because of complexity and single word statistics , i.e, document indexing, are used instead.The IR pioneer Luhn used the term frequency as content descriptive features for documents and it is still today the most used document description method. Using the frequency of co-occurrence words is a promising technique to find word associations in documents. This does not say anything about the nature of the meaning of the associated terms, i.e. , whether they are synonyms, define a term together or are just redundant of common use.Recently, context vectors to expoit these co-occurrence patterns have been implemented in IR.A query is considered a document and is represented in the same way.

- **Ranking** In statistical ranking, the query vector is compared with each document vector.The inner (or dot) product of two vectors is calculated to provide a count of the matching terms. A more common measure of similarity between document and query vectors is the cosine coefficient, in which the similarity betweeen a document in a collection d_j and query q, is described by

$$sim(d_j, q) = \frac{d_j^T \cdot q}{\sqrt{\| \ d_j \ \| \cdot \| \ q \ \|}}$$

The documents are then ranked in decreasing order of similarity and returned.

3 The Vector Space Model

3.1 Introduction to the Vector Space Model

The Vector Space Model was first used in the *SMART*(System for the Mechanical Analysis and Retrieval of Text)system, which was introduced in 1950 (*Salton* and *McGill*, 1997).In the Vector Space Model documents and queries are represented as vectors of terms that occur within documents in a collection. Therefore a list of terms needs to be identified before these vectors can be generated. The value of each document vector element indicates whether a specific term is in a specific document. Therefore a VSM collection is describedd as an n x m matrix, where n is the number of terms, and m is the number of documents in the collection. Often this matrix is referred to as the term document matrix. The terms are extracted from the document collection and queries in a process known as lexical analysis which is explained in the previous chapter. In VSM, a query is treated as a document (an n x 1 matrix) and compared to the document vectors. A document v can be represented by a vector $v = [x_1, x_2, x_3.....x_n]$, where x_i represents each keyword in a collection of documents.If $x_i \geq 1$ then the keyword is present in the document.If $x_i = 0$then the keyword is not present in the document.A query can be represented in the same way.

3.2 Term Weighting

In term weighting, terms which are more important are assigned a higher value than less important terms. The simplest method is to sum the number of times each term appears in a document. The use of term weighting usually results in much better ranking results (*Frakes* and *Baeza-Yates*, 1992). If we refer to a_{ij}as the ij-th element of the term-document matrix A, a weighting scheme will consist of the following three components:

$$a_{ij} = g_i t_{ij} d_j$$

g_i is a global weight which is applied to all non-zero occurrence of term i (all values of row i).t_{ij} is the local weight for term i in document j (*Kolda, 1997*).d_j is a normalisation factor-this may be required as larger documents will tend to receive a higher similarity coefficient, as the term frequencies in that document will be higher. The weighting scheme applied to a term-document matrix and the query vector, is usually specified by a six letter code, that describes the local, global and normalisation values for the term-document matrix and the local, global and normalisation values for the query.In this thesis we have generally used the **lxm.lfx** weighting scheme.

3.2.1 Local Term Weight Formulas

- No local weight (symbol **x**)=**1**
- Term Frequency (symbol **t**)=f_{ij}
- Binary (symbol **b**)=$\chi(f_{ij})$
- Log Weighting (symbol **l**)=**log(**f_{ij}**+1)**

3.2.2 Global Term Weight Formulas

- No global weight (symbol **x**)=**1**
- Inverse Document Frequency (symbol **f**)=**log(**$\frac{n}{\sum_{k=1}^{n} \chi(f_{ik})}$**)**

3.2.3 Normalisation Formulas

- No normalisation (symbol **x**)=**1**
- Normalisation (symbol **n**)=$\sqrt{\sum_{k=1}^{m}(g_k t_{kj})^2}$

4 Latent Semantic Indexing

4.1 Introduction to Latent Semantic Indexing

An important extension of the Vector Space Model is LSI. This method takes advantage of the implicit higher-order structure in the association of terms within a document to improve the number of relevant documents retrieved (*Deerwester, Dumais, Landauer, Furnas and Harshman*, 1990). For example, a person entering a query on *myocardial infarctions* may also retrieve documents on *heart attack*. LSI is a method of dimensionality reduction as the latent semantic space which is created in LSI has less dimensions than the original space (which will have as many dimensions as it has terms) (*Manning and Schutze*, 1990). LSI can be seen as a similarity method, that is an alternative to word overlap measures, such as "td-idf" (*Rosario*, 2000). LSI generally uses a statistical method called **Singular Value Decomposition** (SVD) to uncover the word associations between documents. SVD is a least squares statistical method, and is similar to liner regression. The effect of SVD is too move words and documents that are closely associated near to one another in the projected space. It is possible for an LSI based system to locate terms which do not even appear in a document. Documents which are located in a similar part of the concept space (ie. which have a similar meaning) are retrieved, rather than only matching keywords. By using a concept space, LSI addresses two main problems in IR:

- *Polysemy*, or the problem that most words have more than one meaning, and that meaning is obtained from the word's context. For example, a reference to the word 'bat' would have a different meaning in documents on cricket, or mammals, or home insulation.

- *Synonymy*, or the problem that there are many ways of describing the same object. The presence of synonyms tends to decrease the recall performance of IR systems (*Deerwester et al.*, 1990).

The average precision of LSI over several information science test collections has been reported as up to 30% better that standard keyword vector methods (*Berry, Dumais and O'Brien*, 1995). Disadvantages of LSI include the large amount of storage required for the SVD representation. Retrieval efficiency may not as good as traditional IR, as LSI needs to compare the query against every document in the collection (as opposed to using an inverted index which only needs to examine documents which include the query terms). Another criticism of LSI is that the SVD method is designed for normally distributed data but a term-by-term matrix (even if weighted) from a document collection may not be normally distributed (*Rosario*, 2000). It has been suggested that a dimensionality reduction based on the Poisson distribution would provide a better approximation of the original term by document matrix. When a new document needs to be added to the semantic space, it is not necessary to recompute the SVD of the new term by document matrix. Folding in refers to the process of adding a new document vector into the space. This involves representing the new document as a weighted sum of it's component term vectors and then appending this document vector to the set of existing document vectors. The document vector can then be multiplied with the transpose of the document matrix (*Rosario*, 2000). Another method of adding new documents is SVD-Updating which is a more expensive, but more accurate procedure (*Berry and Browne*, 1999).

4.2 Singular Value Decomposition

Term-Document matrices are usually sparse (usually only around 1% of cells are populated (*Berry et al.*, 1995)). It is more efficient if the most important information from the matrix can be identified and this low-rank approximation is used for matrix storage and query processing. The approximation would reduce the amount of noise in the Term-Document matrix (due to problems such as polysemy and synonymy) (*Berry and Browne*, 1999).The low-rank approximation extract features that are common in the rows and columns of the matrix, which clusters documents for retrieval based on common features (*Kolda and O'Leary*, 2000). This section provides detail of the traditional method for LSI, which is SVD. If we let D be a term by term document matrix defined as

$$D = (\hat{d}_1, ..., \hat{d}_n) \; where \; \hat{d}_j = \frac{d_j}{\parallel d_j \parallel}$$

We can then apply the SVD to convert D into the product of three matrices:

$$D = USV^T$$

where U is an $m \; x \; r$ matrix and V is an $n \; x \; r$ matrix, r is the rank of D and S is a diagonal matrix which contains the singular values (*Berry and Browne*, 1999). By keeping only the k largest singular values in S, and the corresponding columns in U and V, D can be approximated by:

$$D_k = U_k S_k {V^T}_k \; (Deerwester \; et \; al., \; 1990)$$

The D_k matrix should now contain the major associational structure in the matrix and has left out the noise. In this reduced model, the overall pattern of term usage determines how close the documents will be located, regardless of the precise words in the documents

(*Dumais*, 1992). Document query similarity can be measured by calculating the cosine between the document vectors, d_k and a query vector, q_k. This can be achieved by calculating $s = \tilde{q}^T \tilde{A}$, where:

$$\tilde{A} = S_k^{1-\alpha} V^T{}_k$$

and the query vector is projected into the same k-dimensional space by:

$$\tilde{q} = q^T U_k S k^{\alpha}$$

(*Kolda*, 1997) The performance of LSI queries generally improves as k increases, but will decrease past a threshold. This threshold is an open research issue, however the LSI typically performs best with a small number of dimensions compared to the number of unique terms. In addition, the LSI seems to perform well when a major portion of the meaningful structure of the document collection is contained in the smaller set of dimensions (*Dumais*, 1992).

5 Word Space based Random Indexing

5.1 Introduction to Word Space Model & Context Vector

In the standard word space methodology, the high-dimensional vector space is produced by collecting the data in a co-occurrence matrix F, such that each row F_w represents a unique word w and each column F_c represents a context c, typically a multi-word segment such as a document, or another word. In the former case, where the columns represents documents, we call the matrix a words-by-documents matrix, and in the latter case where the columns represents words, we call it a words-by-words matrix. VSM & LSI is an example of a word space model that uses document-based co-occurrences. The cellsF_{wc} of the co-occurrence matrix record the frequency of co-occurrence of word w and document or word c. As an example, if we use document-based co-occurrences, and observe a given word three times in a given document in the data, weenter 3 at the corresponding cell in the cooccurrence matrix. By the same token, if we use word-based co-occurrences and observe that two given words occur close to each other five times in the data, we enter 5 at the corresponding cell. The frequency counts are usually normalized and weighted in order to reduce the effects of high frequency words, and, in case document-based co-occurrences are used, to compensate for differences in document size. The point of the co-occurrence matrix is that the rows F_w effectively

constitute vectors in a high-dimensional space, such that the elements of the vectors are (normalized) frequency counts, and the dimensionality of the space is determined by the number of columns in the matrix, which is identical to the number of contexts (i.e. words or documents) in the data. We call the vectors *Context Vectors*, since they represent the contexts in which words have occurred.In effect, the context vectors are representations of the distributional profiles of words, which means that we may define distributional similarity between words in terms of vector similarity. By virtue of the distributional hypothesis, this makes it very straight-forward to compute semantic similarity between words: we simply compare their context vectors using any of a wide range of possible vector similarity measures, such as the cosine of the angles between the vectors.

5.2 Problem with the Word Space Model

- A majority of the cells in the matrix are zero due to the sparse data problem. That is, only a fraction of the co-occurrence events that are possible in the co-occurrence matrix actually occur, regardless of the size of the data(*Zipf's law*(*Zipf,1949*)).

- Dimension reduction techniques such as SVD tend to be computationally very costly, with regards to both memory consumption and execution time. For many applications, and especially for large vocabularies and large document collections, it is not practically feasible to compute an SVD.

- Dimension reduction is typically a one-time operation, which means that the entire process of first constructing the co-occurrence matrix and then transforming it has to be done from scratch, every time new data is encountered.The inability to add new data to the model is a serious deficiency, as many applications require the possibility to easily update the model.

5.3 Introduction to Random Indexing

As an alternative to LSA-like models that first construct a huge cooccurrence matrix and then use a separate dimension reduction phase an incremental word space model called *Random Indexing*[3],based on *Pentti Kanerva*'s work on sparse distributed representations (*Kanerva 1988, Kanerva et al., 2000, Kanerva et al.*, 2001). The basic idea is to accumulate context vectors based on the occurrence of words in contexts. This technique can be used with any type of linguistic context, is inherently incremental, and does not require a separate dimension reduction phase. The Random Indexing technique can be described as a two-step operation:

- First, each context (e.g. each document or each word) in the data is assigned a unique and randomly generated representation called an *index vector*. These index vectors are sparse, high-dimensional, and ternary, which means that their dimensionality (d) is on the order of thousands, and that they consist of a small number of randomly distributed +1s and -1s, with the rest of the elements of the vectors set to 0.

- Then, context vectors are produced by scanning through the text, and each time a word occurs in a context (e.g. in a document, or within a sliding context window), that context's d-dimensional index vector is added to the context vector for the word in question. Words are thus represented by d-dimensional context vectors that are effectively the sum of the words' contexts.

This means that we can use the Random Indexing procedure to produce a standard co-occurrence matrix F of order $w \times c$ by using unary index vectors of the same dimensionality c as the number of contexts, and then collecting the resulting context vectors in a matrix. Such unary index vectors would consist of a single 1 at a different position for each context, and would thus be orthogonal. By contrast, the d-dimensional

random index vectors are only nearly orthogonal. This means that if we collect the context vectors we produce with Random Indexing in a matrix $F_{w \times d}$, this matrix will be an approximation of the standard co-occurrence matrix $F_{w \times c}$ in the sense that their corresponding rows are similar or dissimilar to the same degree but with $d << c$. In this way, we can achieve the same effect as is done in LSI by the use of SVD: transforming the original co-occurrence counts into a much smaller and denser representation.

5.4 Uniqueness of Random Indexing

- First, it is an incremental method, which means that the context vectors can be used for similarity computations even after just a few examples have been encountered. By contrast, most other word space models require the entire data to be sampled before similarity computations can be performed.

- Second, the dimensionality d of the vectors is a parameter in Random Indexing. This means that d does not change once it has been set, new data increases the values of the elements of the context vectors but never their dimensionality. Increasing dimensionality can lead to significant scalability problems in other word space methods.

- Third, Random Indexing uses .implicit. dimension reduction, since the fixed dimensionality d is much lower than the number of contexts c in the data. This leads to a significant gain in processing time and memory consumption as compared to word space methods that employ computationally expensive dimension reduction algorithms.

6 Contextual Network Graph and Spreading Activation

6.1 Introduction to Contextual Network Graph

A standard step in LSI is the creation of a term-document matrix (TDM), which is essentially a weighted lookup table of term frequency data for the entire document collection. In LSI, this matrix is interpreted as a high-dimensional vector space. An alternative interpretation of this matrix is possible, however, with the TDM representing a bipartite graph of term and document nodes where each non-zero value in the TDM corresponds to an edge connecting a term node to a document node. In this model, every term is connected to all of the documents in which the term appears, and every document has a link to each term contained in that document. The weighted frequency values in the TDM correspond to weights placed on the edges of the graph. We call this construct a *contextual network graph*[4]. As an example, consider the miniature document collection in table 1, and its associated term list in table 2. Each edge in the graph has a strength assigned to it whose magnitude depends on our choice of the local and global term weighting scheme used in generating the TDM. The only constraint on weighting schemes is that all edge weights must fall in the interval (0,1).

Table 1. A sample document collection with associated node labels

Node	Document content
1	*Glacial ice often appears blue.*
2	*Glaciers are made up of fallen snow*
3	*Firn is an intermediate state between snow and glacial ice.*
4	*Ice shelves occur when ice sheets extend over the sea.*
5	*Glaciers and ice sheets calve icebergs into the sea.*
6	*Firn is half as dense as sea water.*
7	*Icebergs are chunks of glacial ice under water.*

Table 2. A labeled term list derived from the collection in table 1. All terms occur across at least two documents in the parent collection.

Node	Term	Occurrence count
a	glacial ice	3
b	ice	5
c	glacier	2
d	snow	2
e	firn	2
f	ice sheet	2
g	sea	3
h	water	2
i	iceberg	2
j	sheet	2

We can represent this collection of terms and documents as a contextual network with the topology shown in figure 1.

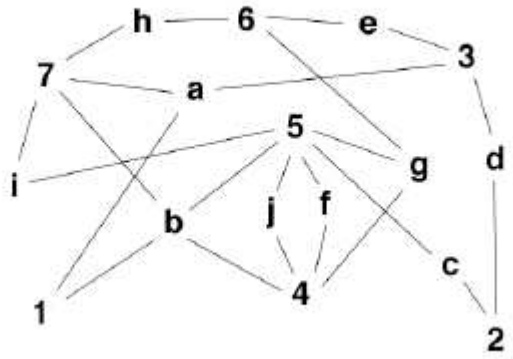


Figure 6.1: Sample contextual network graph indicating connections between the documents in table 1 and content terms in table 2

```
1  procedure energize( energy E, node n_k ) {
2       energy(n_k) := energy(n_k) + E
3       E' := E / degree of  n_k
4       if ( E' > T ) {
5            for each node n_j in N_k  {
6                 E" := E' * e_jk
7                 energize( E", n_j )
8            }
9       }
10 }
```

Figure 6.2: Algorithm for spreading activation

6.2 Introduction to Spreading Activation

We can search the collection represented by this graph by energizing a query node and allowing the energy to propagate to other nodes along the edges of the graph based on a set of simple rules. The total energy deposited at any given node in the graph will depend both on the number of paths between it and the query node, as well as the relative strength of the connections along those paths. This corresponds to the intuition that documents that share many rare terms are likely to be semantically related. It also enables the same kind of enhanced recall provided by LSI, since a query on a particular keyword may still reach a document that does not contain the word itself, but is closely linked to other documents that do.

Because the initial energy dissipates as it spreads over the graph (the requirement that energy dissipate is the reason for the constraint on edge weights), results for any search are localized to a single region of the graph, with important implications for scalability.

In the example above, a search on *iceberg.* would begin by activating the node corresponding to the query term, in this case node i. This query node is assigned a default starting energy E, which is distributed to neighbor nodes according to the following algorithm shown in Figure 6.2.

In this description of algorithm N_k is the set of all neighbor nodes of n_k, e_{jk} is the

weight of the edge connecting nodes n_j and n_k, T is a constant threshold value, and energy(n_k) is a data structure that stores node energy values for the duration of a query. Note that this version of the algorithm performs a depth-first traversal of the graph. In the case where N_k is sorted by decreasing edge weight, the traversal is also best-first. In this thesis we have implemented both the depth-first and breadth-first (best first) versions of this algorithm.In our example, the query consists of a single term, and the search terminates after the energy from the initial node has been distributed as far as it can go before dipping below the threshold T. In the case of queries consisting of multiple nodes, the procedure would be repeated for each query node in turn, with the final energy values for nodes in the graph a superimposition of the individual searches.

Once all the nodes in a query have been processed, results are gathered in a collection step, with nodes sorted by reverse order of accumulated energy.

7 Contextual Vector Approach

7.1 The Retrieval Model

In this approach a word which is assumed to have a dominant meaning in a document (*Gale et al., 1992*), is represented in the form of a context vector.The semantic closeness of two words is indicated by the distance between their context vectors.

In the retrieval stage, the documents and the queries are indexed using these context vectors and the traditional vector model as in SMART system is used to compute similarity and rank the documents.

7.2 Computing the Context Vector

To compute the context vector for a target word, all candidate words which can possibly be included in the context vector are first collected. This is accomplished by extracting all the words which appear at least once in the local contexts of the target word in the document.In our work, a window of K words (K words on either side of the target word) is considered as local context.Then a weight is assigned to each collected word based on the following formulae:

$$F_r(I \mid T)/F_r(T)$$

the frequency of the word I appearing in the window with the target word T divided by the term frequency of the target word.

The context vector is then normalized. As a result, each of the words in the context vector will acquire a weight between 0 and 1. The more frequently a word co-occurs with the target word in the local contexts, the larger the weight.

8 System Evaluation Approach

8.1 Introduction

The following six qualities have been identified as good measurements of IR systems (*van Rijsbergen*, 1979):

- The coverage of a collection (the amount of relevant material).
- The response time between entering a query and receiving a response from the IR system.
- The effectiveness of the output display (for example, documents ranked in descending order).
- The effort required by a user to obtain answers to their search request.
- The proportion of relevant material actually retrieved in the system output (recall).
- The proportion of retrieved material that is relevant (precision).

This chapter reviews recall and precision measurements and then examines the two reference collections MED and NPL we have used.

8.2 Recall and Precision

The two main measurable statistics used are recall and precision. The IR literature generally uses interpolated average precision, non-interpolated average precision, R-Precision and an average precision from three recall levels (25%, 50% and 75%). Average precision versus recall figures are a standard evaluation strategy for IR systems.

Recall, R, is a measure of how much information is retrieved by the search. It is a ratio of relevant documents retrieved for a given query over the number of relevant documents in the document collection, and is defined by:

$$R = D_r/N_r,$$

where D_r is the count of relevant documents returned and N_r is the count of all relevant documents which could be returned. Generally, 11 recall levels are presented in the literature: 0%, 10%, 20%, ..., 100%. For the recall measure to be useful the number of relevant documents needs to be known, which is not always possible (for example, in an Internet search engine where the number of documents is dynamic).

Precision, P, is the ratio of the number of relevant documents retrieved from the search over the total number of retrieved documents and is defined by:

$$P = D_r/D_t,$$

where D_r is the count of relevant documents returned and D_t is the count of all documents returned.

Generally, as precision increases, recall decreases.

8.3 Average Precision (25,50,75)

Average precision is often defined in the literature as precision measured at 25%, 50% and 75% recall. This measure is simple to calculate, and has been included in this thesis for comparison purposes with other studies.

$$(p_{25} + p_{50} + p_{75})/3$$

8.4 Interpolated Average Precision

Usually 11 point interpolated precision is used. The maximum precision between each recall level is averaged across the 11 levels. The interpolated value for recall level 0 is the highest precision between 0 and 10% recall. And similarly for levels 10% ... 90%, the highest precision between 10% - 20% ... 90% - 100% are used. The value for 100% is the same as the R-Precision value.In this thesis we have used this as the primary evaluation criterion.

8.5 Non-Interpolated Average Precision

Non-Interpolated average precision is the average of the precision values for a query. This is calculated by summing the precision value as each relevant document is returned and dividing this sum by the number of relevant documents.Non-Interpolated Average precision may be a less useful measure than Interpolated Average Precision, as the recall levels for each query might be distinct from the 11 standard recall levels.

8.6 R-Precision

This measure reflects the precision when all relevant documents have been returned. R-Precision is a useful way of testing different algorithms over individual queries. The average R-Precision can be calculated for a set of queries, however this may not be precise.

8.7 Reference Collections

A reference collection contains a set of documents, a set of questions and a list of documents which are relevant to the questions. Deciding which documents are relevant requires independent judges to assign relevance values to documents. How relevant a document is to a particular query is subjective, however the difference between user perceptions is not large enough to invalidate experiments which have been made with document collections which have questions and lists of relevant documents for these questions.

We have used the MED and the NPL datasets for the comparsion of various approaches introduced in previous chapters.These datasets were downloaded from the SMART web-site at Cornell University.One of the principal reason for choosing more than one dataset is to emphasize and generalize our results in all alternative test document collections.

The Medline (also referred to as MED, MEDLARS or MED1033) collection is a small test collection of 1033 biomedical abstracts and 30 queries which has been used frequently in the IR literature. Because of it's size Medline is a useful collection for the preliminary testing of IR algorithms (*Baeza-Yates and Ribeiro Neto*, 1999). The NPL (also referred to as VASWANI) collection is test collection of 11429 documents from the National Physical Laboratory and 93 queries.

Medline document 501 is shown in Figure 8.1. The text following .I is the document

```
 .I 501 .W 3107.  studies on y-crystallin from calf lens.  ii.
purification and some properties of the main protein components
four proteins belonging to the y-crystallin group were purified by
chromatography on sulphoethyl-sephadex and phosphate-cellulose columns.
the proteins were homogeneous in gel and immunoelectrophoresis experiments
and could be crystallized.  their molecular weights, n-terminal amino
acid sequences and antigenic structures were all similar, but their amino
acid compositions and the sulphydryl groups contained showed certain
dissimilarities.  it is probable that the 4 proteins possess small
differences in their primary structure, which are not associated with the
antigenic sites and which may have arisen from mutations during evolution.
```

Figure 8.1: Text for document 501 in the Medline collection

```
 I 1 W
the crystalline lens in vertebrates including humans
```

Figure 8.2: Query 1 (of 30) from the Medline collection

number. The meaning of the .W is unclear.The first question in the set of 30 is displayed in Figure 8.2, along with the list of relevant documents in Figure 8.3. Note that document 501 in Figure 8.1 is considered a match for the query in Figure 8.2.

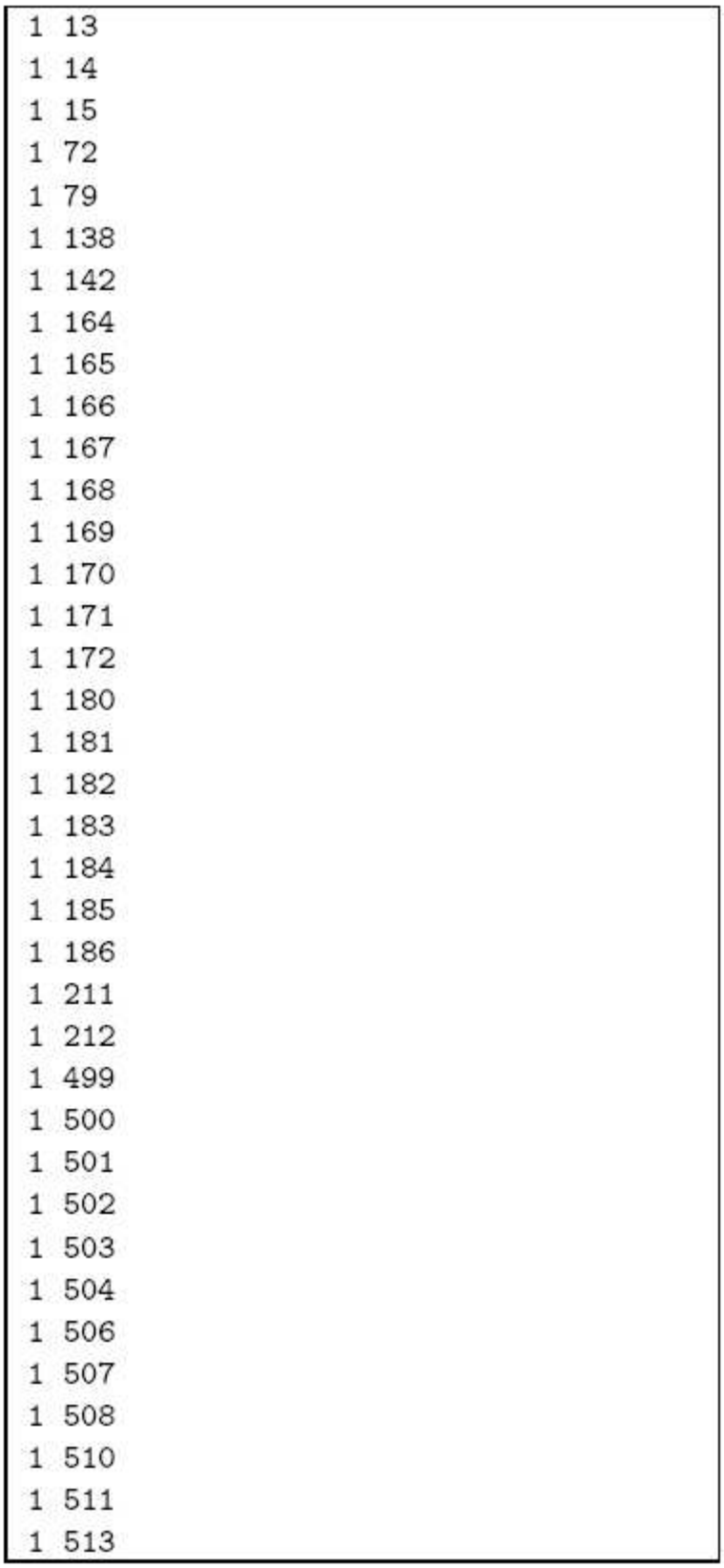


Figure 8.3: Relevant documents from the MED collection for query 1

9 Experimental Results & Discussion

We present our experimental results in this chapter.We have used the 11-point interpolated average Recall-Precision relationship for our comparison study. .

9.1 Vector Space Model Results

An average interpolated precision of **0.5227** and **0.2056** was obtained for this approach for the Medline and NPL datasets.Figure 9.1 & 9.2 shows the corresponding 11-point precision-recall curves.

9.2 Latent Semantic Indexing Results

A maximum average interpolated precision of **0.558** is obtained at a K value of around 30 for the Medline dataset.Figure 9.3 shows a plot of average interpolated precision against values of K values from 10 to 100.This curve gains a maxima at around K=30.(Please note that K is the reduced dimension in the LSI approach)

9.3 Comparsion of VSM & LSI Results

VSM performs better at lower recall levels of 0.0 and 0.1 as compared to the LSI approach after which LSI shows better precision.Overall improvement of LSI at K=30 is **6.77%**

Figure 9.1: Medline 11-point precision-recall curve

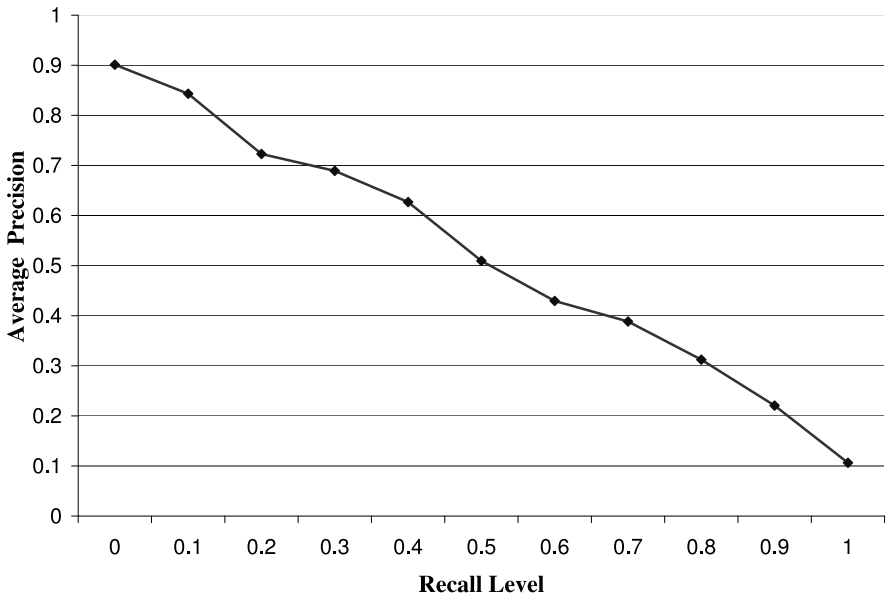


Figure 9.2: NPL 11-point precision-recall curve

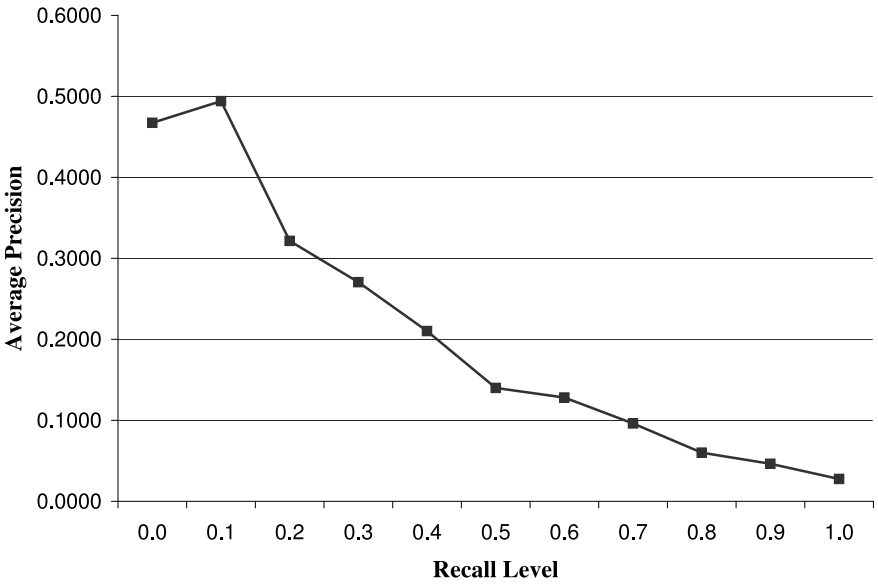
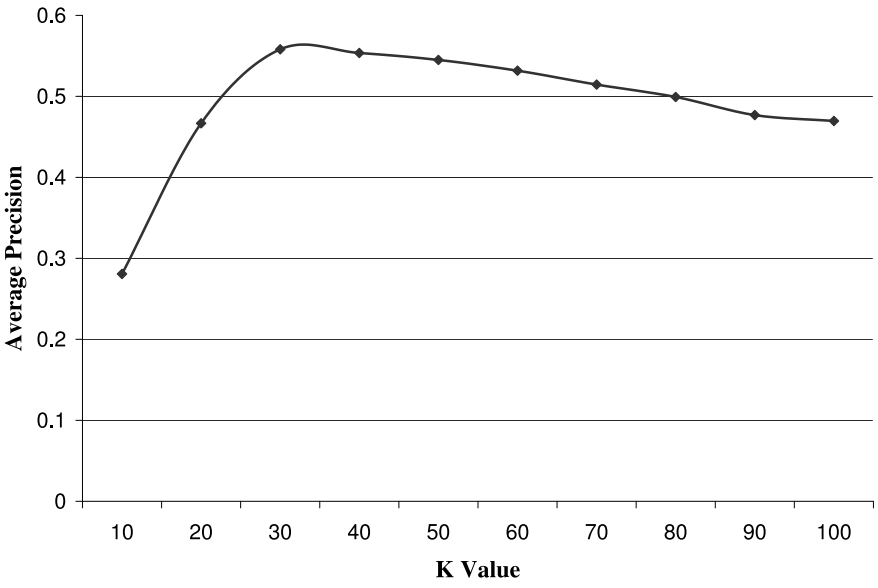


Figure 9.3: Medline average interpolated precision versus K-value curve



over the VSM approach.Figure 9.4 shows the corresponding plots.

9.4 Random Indexing Results

We have obtained an average interpolated precision value of **0.5552** which is an improvement of **6.227%** over the VSM approach with this approach (we took an index vector of size 3000,context window of size 10,number of 1(s) in the index vector equal to 25) for the Medline datasets.Figure 9.5 compares the average precision values obtained at various recall levels in this approach with the VSM approach. Figure 9.6 shows a comparison between the LSI at K=30 and Random Indexing (3000,10,25) results.Before recall level of **0.3** random indexing performs better and at lowerer recall LSI takes the lead.

9.5 Contextual Vector Approach Results

With this approach we have obtained an average interpolated precision value of **0.5507**which is an improvement of **5.371%** over the VSM approach.This precision value was obtained for the context window size of **5**.Figure 9.7 shows the variation in the average interpolated precision value with the context window size. Figure 9.7 shows the comparison of results of this approach with the random indexing results recall level wise.

9.6 Spreading Activation Results

The contextual betwork based spreading activation approach poorly performs on both the Medline and NPl datasets.Also no literature study gives its performance statistics.We tried both the dfs- and bfs-based activation approaches but no improvement in performance even with respect to VSM approach was achieved.

Figure 9.4: Comparison of LSI (K=30) and VSM results for Medline dataset

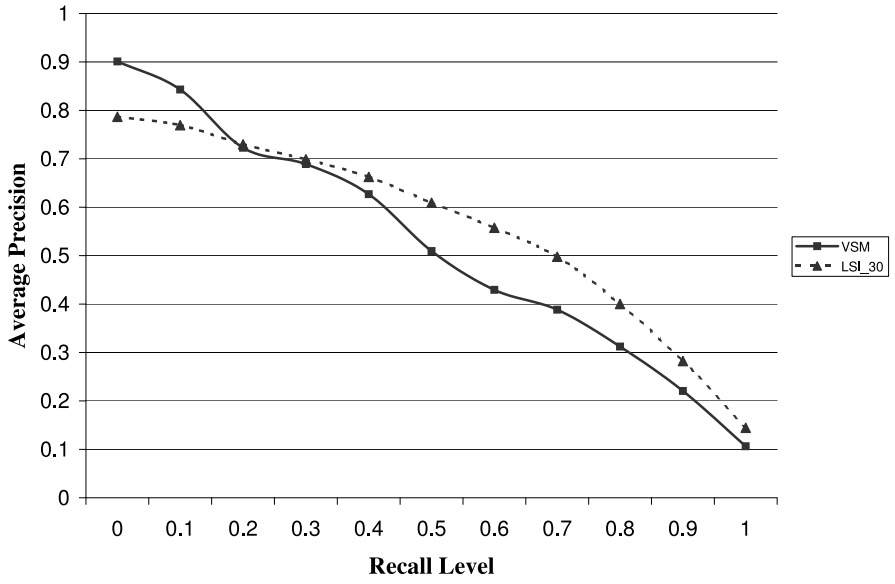


Figure 9.5: Comparison of Random Indexing and VSM results for Medline dataset

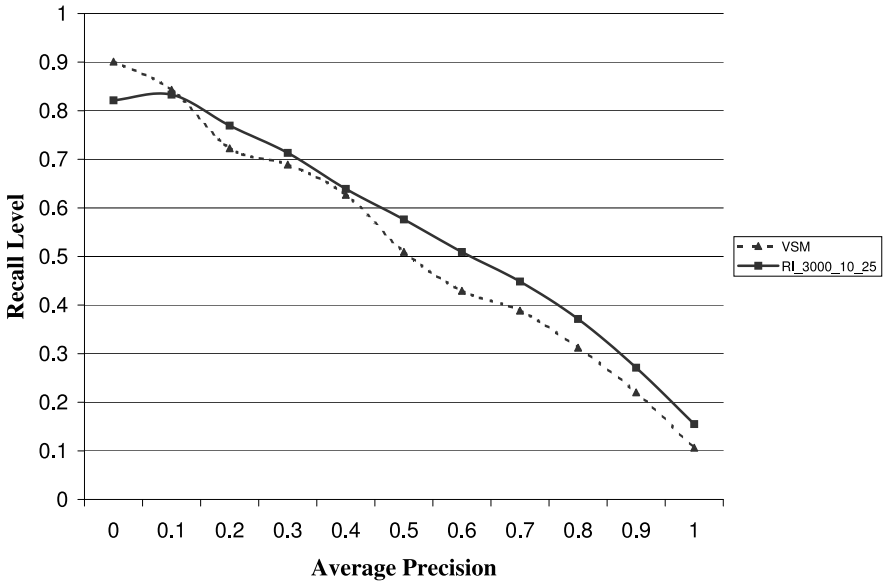


Figure 9.6: Comparison of Random Indexing and VSM results for Medline dataset

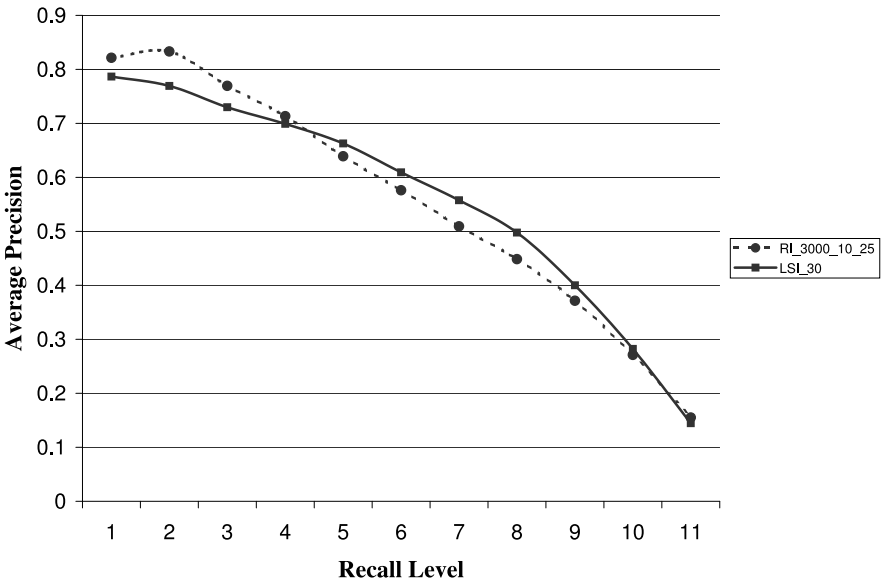


Figure 9.7: Average Interpolated Precision versus the context window size

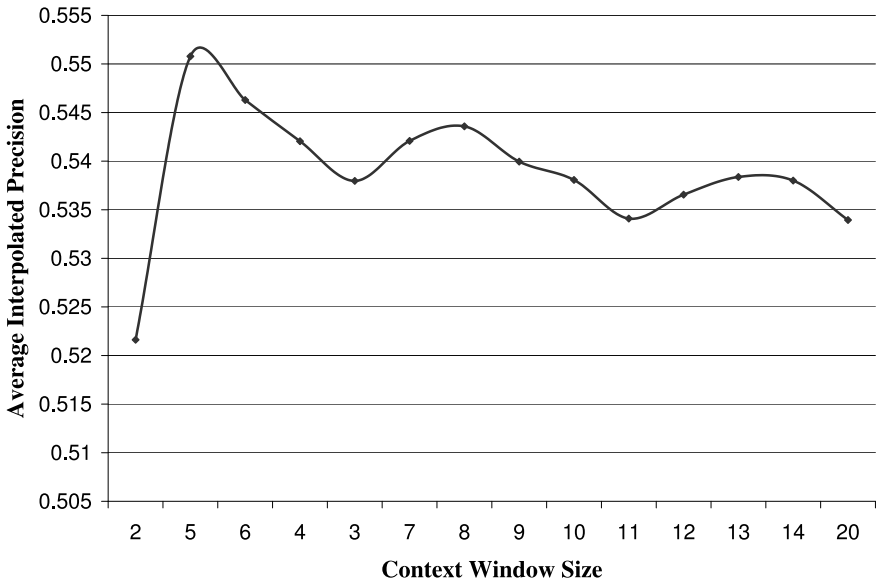
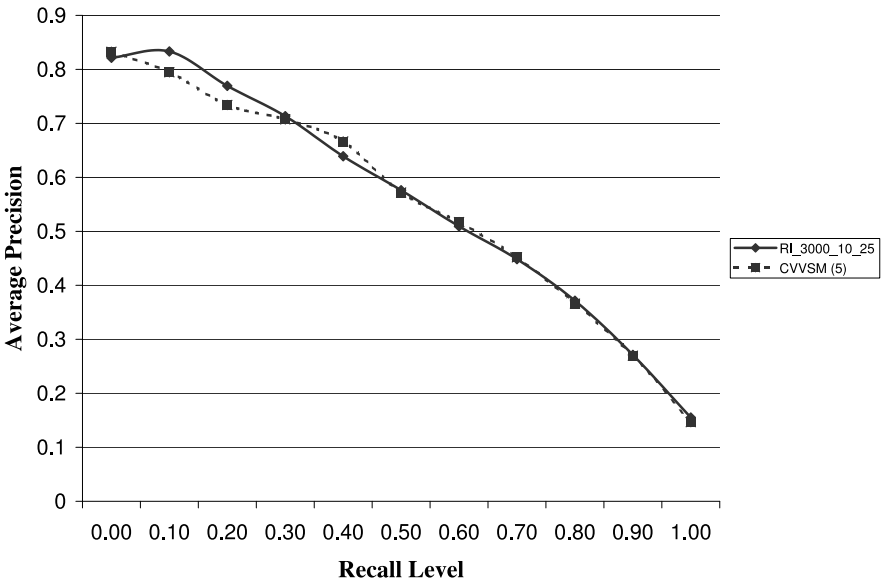


Figure 9.8: Average Interpolated Precision versus the context window size



9.7 Discussion

VSM performs better than any other approach at lower recall levels as **0.0** and **0.1**.The other approaches including random indexing, LSI and contextual vector based perform better at higher recall levels.Random indexing due to its computational less expenisive nature and peformance at par with LSI approach can be a better alternative.Contextual Vector based approach in comparison to random indexing performs better at lower recall level and performs at par at higher recall levels.Further study can involve a combination of these two approaches to enhance performance.

Bibliography

[1] G.Salton, A.Wong and C.S. Yang

A Vector Space Model for Automatic Indexing.

Communications of the ACM, vol. 18, nr. 11, pages 613-620

[2] S. Deerwester, Susan Dumais, G. W. Furnas, T. K. Landauer, R. Harshman

Indexing by Latent Semantic Analysis.

Journal of the Society for Information Science 41 (6): 391-407

[3] Kanerva, P., Kristoferson, J. & Holst, A.

Random Indexing of Text Samples for Latent Semantic Analysis.

Proceedings of the 22nd Annual Conference of the Cognitive Science Society, p. 1036. Mahwah, New Jersey: Erlbaum, 2000

[4] Maciej Ceglowski, Aaron Coburn, and John Cuadrado

Semantic Search of Unstructured Data using Contextual Network Graphs.

National Institute for Technology and Liberal Education Middlebury College, Middlebury, Vermont, 05753 USA

[5] Jing, H., Tzoukermann, E

Information retrieval based on context distance and morphology.

Proceedings of the 22 nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '99), Seattle, WA (1999) 90–96.

[6] Medline Collection

Wikipedia The Free Encyclopedia

ftp://ftp.cs.cornell.edu/pub/smart/

[7] Eric Schmidt and Mike Lesk

Wikipedia The Free Encyclopedia

[8] Gerard Salton

Wikipedia The Free Encyclopedia

[9] Preece, Scott.

A spreading activation network model for information retrieval

PhD thesis, CS Dept., Univ. of Illinios, Urbana, IL. 1981.

